**Experiment Number:** 05

**Aim:** Tutorial on lists in Haskell.

**Lab Objective:** Design and implement declarative programs in functional and logic programming languages

**Lab Outcome Mapped:** Design and Develop solution based on declarative programming paradigm using functional and logic programming(LO2)

**Requirements:** Any text editor to be able to edit Haskell code and Glasgow Haskell Compiler 8.0+ version.

**Performance:**

[**Note: As part of this experiment students need to understand lists in Haskell. In order to do so students will step by step follow the tutorial in ghci and write the output obtained in the correct place. Once all outputs are entered you can convert the word file to pdf and upload it.**]

In Haskell, lists are a *homogenous* data structure. It stores several elements of the same type. That means that we can have a list of integers or a list of characters or a list of floats or strings but we can't have a list that has combinations of different types.

1. To begin with, we will create list of Int data of size 4:



Note: let keyword is used only with ghci i.e. in the interpreter environment to create a variable with a particular name and use it in the session.

1.a. Now u create a list of 5 Float numbers with name x and print its values

Prelude> let x=[4.5,6.0,5.0,4.6,6.8778]

Prelude> x

[4.5,6.0,5.0,4.6,6.8778]

1.b. Now u create a list of 3 chars with name y and print its values

Prelude> let y=['f','d','f']

Prelude> y

"fdf"

1.c. Now u create a list of 3 Strings with name **myname** that contain your first name, middle name and last name and print its values

Prelude> myname=["Anup","Shrikant","Kunte"]

Prelude> myname

["Anup","Shrikant","Kunte"]

2. Execute these command and write down output (Learn use of ++ (concatination operator))

2.a. Prelude>[1,2,3,4] ++ [9,10,11,12]

**Output :** [1,2,3,4,9,10,11,12]

2.b. Prelude> "hello"++" "++"world!"

**Output:** "hello world!"

2.c. Prelude> [‘w’,’h’] ++ [‘a’,’t’]

**Output :** "what"

3. Learn how to work with **cons** (:) operator

3.a. Prelude>'A':" SMALL CAT"

**Output:** "A SMALL CAT"

3.b. Prelude>1:[4,6]

**Output:** [1,4,6]

3.c. Prelude>1:4:6:[]

**Output :** [1,4,6]

3.d. Prelude>1:4:6

**Output:** Error

4. Indexing operator (!!) Note output o below commands and answer following:

4.a Prelude > [9.4,33.2,96.2,11.2,23.25]!!3 Output : 11.2

4.b Prelude > [9.4,33.2,96.2,11.2,23.25]!!0 Output : 9.4

4.c Prelude > [9.4,33.2,96.2,11.2,23.25]!!5 \*\*\* Exception: Prelude.!!: index too large

Based on the above outputs !! operator is \_\_\_\_ ary operator first argument is \_\_\_\_\_\_ and second argument is \_\_\_\_\_\_\_ . In haskell, the index starts with \_\_\_\_\_\_\_ and ends in \_\_\_\_\_\_.

5. List of lists : In haskell one can create list of lists

5.a Prelude> let x=[1,2,3]; y=[5,6]

5.b Prelude> x

5.c Prelude> y

5.d Prelude> let z=[x,y]

5.e Prelude> z

5.f Prelude> :type x

5.g Prelude>:type y

5.h Prelude>:type z

Note: :type displays type signature of an object

Based on the above output z has type signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ means it's a list of \_\_\_\_\_\_\_\_.

6. Now we will make a table of basic functions on list you need to fill in missing entries:

| Sr. | Name | Example | Output | Type signature |
| --- | --- | --- | --- | --- |
| 1 | head | head [4,2,5,6] | 4 | head :: [a] -> a |
| 2 | tail | tail [5,4,3,2,1] | [4,3,2,1] |  |
| 3 | last | last [5,4,3,2,1] | 1 |  |
| 4 | init | init [4,2,5,6] | [4,2,5] |  |
| 5 | length | length “hi there” | 8 |  |
| 6 | null | null [] | True |  |
| 7 | reverse | reverse [5,6,3] | [3,6,5] |  |
| 8 | take | take 3 [5,6,2,3,4,3] | [5,6,2] |  |
| 9 | drop | drop 2 [4,5,2,2,4] | [2,2,4] |  |
| 10 | elem | 4 `elem` [3,4,5,6] | True |  |
| 11 | maximum | minimum [8,4,2,1,5] | 1 |  |
| 12 | minimum | maximum [1,9,2,3,4] | 9 |  |

7. Understand how to specify a range while creating list

7.a. Prelude> let alphabet=[‘a’..’z’]

7.b. Prelude> alphabet

7.c. Write range of number to create list of first 20 even numbers

7.c Write range of number to create list of multiple of 4 starting at 8 upto 45

7.d Write output for following

Prelude> take 24 [13,26..]

In this example the take function has a \_\_\_\_ number of arguments last of which is a list. It is a \_\_\_\_\_\_

(choose either finite or infinite) type of list. This statement executes perfectly owing to the fact that Haskell functions follow \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (eager/lazy) evaluation semantics.

8. Execute command take 10 (cycle [1,2,3]) on ghci and explain the output.

9. Execute command take10 (repeat 5)

10. For Following list comprehension write the output list that it represents:

| sr. | List Comprehension | Elements of list |
| --- | --- | --- |
| 1 | [x^2 | x <-[1..5]] |  |
| 2 | [x\*2 | x<-[6..12],x\*2<=15] |  |
| 3 | [x|x←[50..100], x `mod` 7==3] |  |
| 4 | [x|x<-[10..20],x/=13,x/=15,x/=19] |  |
| 5 | [x\*y|x←[2,5,10], y←[8,10,11], x\*y>50] |  |
| 6 | [x+y|x<-[2,5,10],y<-[8,10,11]] |  |
| 7 | [(a,b,c) |c<-[1..10],b<-[1..c],a<-[1..b],a^2+b^2==c^2] |  |

Conclusion: Thus we have learned about List data structure in haskell.

Reference:

[1] Learn you Hakell for greate good. A biginner’s guide, <http://learnyouahaskell.com/>, Accessed on 02/12/2020